Table of Contents

Introduction

In March 2016, AlphaGo beat Lee Sedol in a five-game match, the first time a computer Go program has beaten a 9-dan professional without handicaps. Artificial intelligence shocked people all over the world again. One of the key reasons AlphaGo can beat human is that it can predict future situations longer than human brain in the game board. Under some certain rules, computer can search every possibility and find the best solution to react. This agent has one evaluate function to evaluate each possible position. The agent will use algorithms to try each position, and simulate future board situation about next possible steps. This way helps the agent to evaluate and choose current step reasonable. In conclusion, the key point of the AI is how to find the best steps based on current situation and future possibility.

Reversi (Othello) is a classical strategy board game for two players, played on an 8×8 uncheckered board. There are sixty-four identical game pieces called *disks* (often spelled "discs"), which are light on one side and dark on the other. Players take turns placing disks on the board with their assigned color facing up. During a play, any disks of the opponent's color that are in a straight line and bounded by the disk just placed and another disk of the current player's color are turned over to the current player's color. The board will start with 2 black discs and 2 white discs at the center of the board. They are arranged with black forming a North-East to South-West direction. White is forming a North-West to South-East direction. You can flank any number of discs. You may capture discs vertically, horizontally, diagonally. You may capture in multiple directions at the same time. All discs that can be flipped must be flipped. You cannot pick and choose which ones are captured. If any of the agents cannot make any further move then the opponent will win. Initial state of the game is shown in figure.

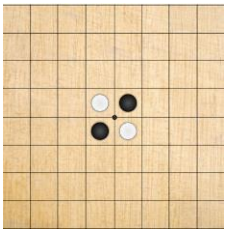


Figure 1: Initial state of the game

The game of Reversi was invented in 1883. A modern Mattel game, published as Othello and with a change to the board's initial setup, was patented in 1971.

Objectives

The objective of the game is to have the majority of disks turned to display your color when the last playable empty square is filled. Each player counts the number of spaces occupied by their color. Highest count is the winner. Games can end before all spaces are filled. And when the player don't have any moves to play then also opponent wins.

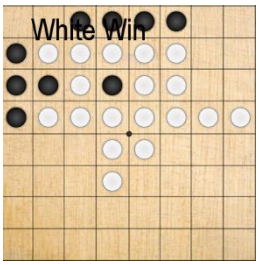


Figure 2: A white player win situation

Agent Description

Performance Measure:

i. Win rate of AI with different algorithms

ii. Time taken to make a decision

Environment:

i. Game Board

ii. AI's and Player's Checker

Actuators:

Game Playing Function and algorithms

Sensors:

Knowing the state of the board

Agent Environment

The following table shows the agent environments with the description

Environment		Description
Deterministic/Stochastic	Deterministic	The agent knows with certainty the next state of the environment and the resulting state doesn't change given the same state and action at all times.
Episodic/Sequential	Sequential	The environment is sequential as the past history of actions in the game affects the current decision which further affects all the future decisions.
Static/Dynamic	Static	Once its agents turn, program will wait until agent makes it move and vice versa.
Discrete/Continuous	Discrete	There are a finite number of actions to be performed. Example: Initially we have 2 moves to perform.
Single agent/Multi agent	Multi agent	Contains two agents, an AI agent and a Human player. Both target to maximize their color turned.
Fully/Partially observable	Fully	The agent has access to full information of the environment so it knows the state of the game at each step to make optimal decision.

Problem specification

A game for two players, given an 8*8 uncheckered board and sixty-four identical games pieces called disks and the goal is to have the majority of disks turned to display your colour when the last playable empty square is filled or there is no any legal move to perform either by AI agent or a human player agent.

This program consists of UI-display part, game-play part and AI-agent part.

In the UI-display part: Using pygame (3-rd game lib) to build UI and proceed the user action.

In the game-play part: Checking the steps are legal or not, the available reverse and handle the reverse. After one player finish the steps and change the board. It will send data to the UI-display part in order to update the UI.

In the AI-agent part: AI can make a reasonable step decision, after implemented evaluation function and algorithms.

State space representation/ data source

The state-space complexity of a game is the number of legal game positions reachable from the initial position of the game. Any game has a well-defined state or position at upcoming moves. The starting position is already defined and the next state are the possible states that a player can move in the 8*8 table and according to the move of human agent, AI moves the black disc according to the algorithm. The figure below shows the possible state for the human agent at first move.

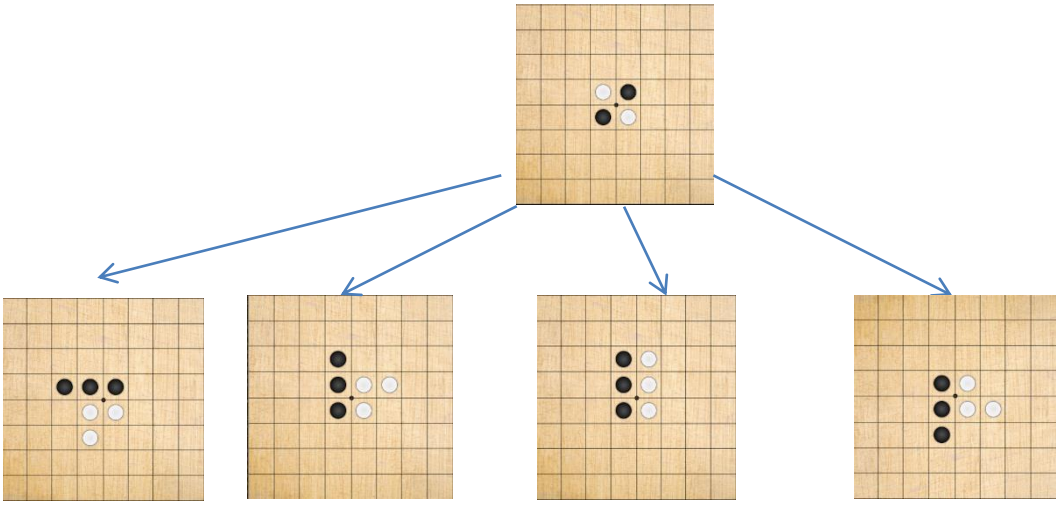


Fig: First (Initial) state and its possible states

There will be three algorithms and one evaluates function to help the agent to make a decision. The evaluate function tells whether the position is good or not, according to some common sense. For example, the corner is the best position to put disks, because the opponent cannot traverse one in the corner. The three algorithms are Greedy algorithm, Min-Max algorithm and Alpha-Beta algorithm. Greedy algorithm don't use the evaluate function, it only focus on how to reverse more disks. Min-Max algorithm and Alpha-Beta algorithm will use evaluate function to calculate positions score in order to get the best moving position.

Complexities of the game are shown in table:

Game	Board size	State space complexity(as log to base 10)	Game tree complexity(as log to base 10)	Average game length(piles)
Reversi	64	28	58	58

Outcome analysis

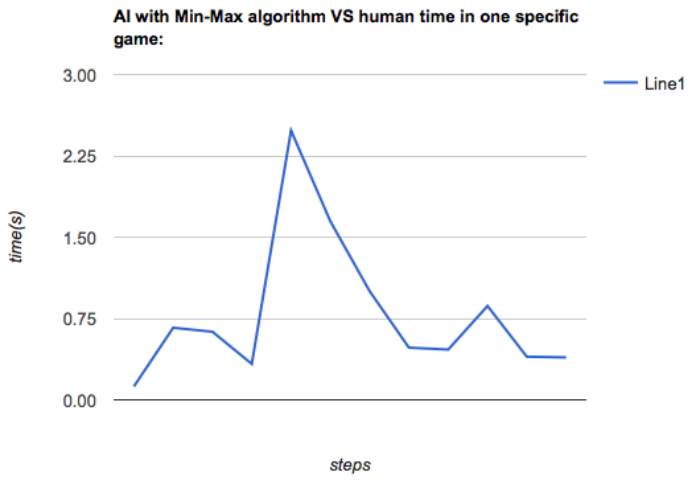
Result of AI with Greedy algorithm (no evaluation function) VS human:

time / result	1	2	3	4	5	6	7	8	9	10
win	√	√	x	√	x	x	√	x	x	x
lost	x	x	√	x	√	√	x	√	√	√

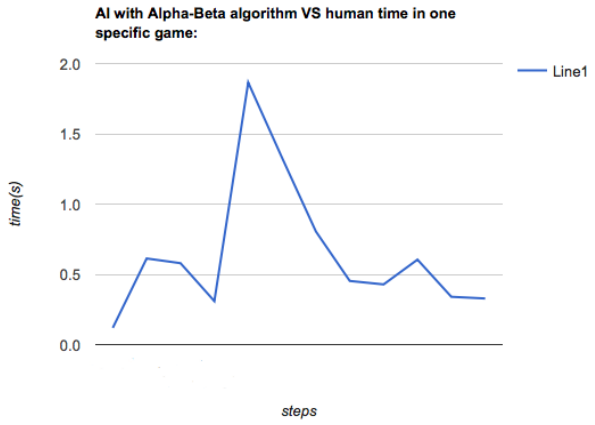
Result of AI with Min-Max algorithm VS human:

time / result	1	2	3	4	5	6	7	8	9	10
win	√	√	√	√	x	x	√	√	√	x
lost	x	x	x	x	√	√	x	x	x	√

Time of AI with Min-Max algorithm VS human time in one specific game:



Time of AI with Alpha-Beta algorithm VS human time in one specific game:



Conclusion

Using the Greedy algorithm:

The agent's reaction is fast, but only focus on current situation. It cannot beat human often. What's more, it's reaction can be predict by human ,because human can easily understand the Greedy algorithm and learn how to find the pattern to defeat the agent. The result is the Greedy algorithm is a simple and week strategy to build AI agent. We need more powerful way.

Using the Min-Max algorithm and Alpha-Beta algorithm:

Min-Max algorithm and Alpha-Beta use the same evaluation function. The difference between the two algorithms is the second one has the prune function. So it can save much time in the search Tree. As a result, we can see even the decision the two algorithms made are the same. But the time they used are huge different. This mean the prune function actually improved the efficiency in Alpha-Beta algorithm.

Improve aspects:

The evaluation function only uses a simple rule to determine the value of different position in board. But as the game is playing, the value of the position will be different. Even the corner and side are good position. But the agent need merge more reality situation. So the evaluation function can be improved, like the evaluating rule can be change in the game according to current situation.

Also, the agent can add mobility evaluation in the decision part. Because, the mobility can also influence the search space. And influence the decision. The agent should try to create more mobility chance in the game. Not only focus on the position. More information can be found in Michael Buro's published "An Evaluation Function for Othello Based on Statistics".